SYSTEM AND METHOD FOR AN INTEROPERABILITY FRAMEWORK

Inventors: Anna M. Arellano-Payne, Anthem, Arizona
               Dan H. Toraason, Peoria, Arizona
               Adesh H. Gokhale, Pune, India
5   Neeraj Sharma, Haryana, India
               Harish Bharti, Phoenix, Arizona
               Linda Hiles, Ft. Lauderdale, Florida
               Supratim Banerjee, Boca Raton, Florida
               Ana M. Muniz, Pembroke Pines, Florida

10   ## CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority from U.S. Provisional Patent application

serial number 60/280,806, filed April 2, 2001.

BACKGROUND OF THE INVENTION

15   1)   Field of the Invention

This application generally discloses a method of accessing information

and, more particularly, a computer-implemented method of accessing and

exchanging information among otherwise incompatible platforms.

        2)   Background

20   Computer software has typically been operable on only the specific

platform for which the software was created. For example, software developed

for an Apple Macintosh computer is typically operable only on Apple Macintosh

computers and would not be operable on a computer running LINUX. The same

may also be true for the data stored by the various programs, as the data is

25   generally not interchangeable among different platforms. Thus, if one wanted to

share programs and data among various incompatible platforms, for example

exchanging data regarding a particular item, the programs and data were first converted to be operable on the platforms.

In the past, data has been stored and maintained on various different platforms. For example, certain entities use systems based on Microsoft technology, such as Windows NT, Internet Information Services ("IIS"), or Component Object Model ("COM"). Other entities may use systems based on a UNIX platform, such as those developed by Sun Microsystems, using programs such as WebSphere Application Server ("WAS") and programming languages such as JAVA. In certain situations, a single organization may use two different platforms (e.g., different divisions of a single organization may use different platforms). In other situations, an organization may wish to share data and programs with another organization, which uses a different platform.

With the recent proliferation of the use of the Internet and the corresponding growth in connectivity between computers, it has become more desirable to share data and programs among variously distributed locations. Of particular interest is the ability to access various pieces of information through an Internet interface, even when the data and programs containing the information were created for incompatible platforms. In the past, there have been tools used to convert programs such that they are operable on other platforms. However, such tools tended to be cumbersome, expensive, and wasteful in the use of computing resources. Therefore, there is a need for a tool that allows the use of various programs and data across various platforms.

BRIEF SUMMARY OF THE INVENTION

A method of the present invention includes two computing platforms coupled to an interoperability framework. The interoperability framework may be configured to translate information requests from one computing platform into a specific format that can then be translated into a format readable by the second computing platform. The second computing platform can then process the request and supply the requested information to the interoperability framework, which would then translate the information into a format readable by the first computing platform.

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS

The present invention may be described herein in terms of functional block components, screen shots, optional selections and various processing steps. It should be appreciated that such functional blocks may be realized by any number of hardware or software components configured to perform the specified functions. For example, the present invention may employ various integrated circuit components, *e.g.*, memory elements, processing elements, logic elements, look-up tables, and the like, which may carry out a variety of functions under the control of one or more microprocessors or other control devices. Similarly, the software elements of the present invention may be implemented with any programming or scripting language such as C, C++, Java, COBOL, assembler, PERL, Visual Basic, or the like, with the various algorithms being implemented with any combination of data structures, objects, processes,

routines or other programming elements. Further, it should be noted that the present invention may employ any number of conventional techniques for data transmission, signaling, data processing, network control, and the like.

It should be appreciated that the particular implementations shown and
5   described herein are illustrative of the invention and its best mode and are not intended to otherwise limit the scope of the present invention in any way. Indeed, for the sake of brevity, conventional data networking, application development and other functional aspects of the systems (and components of the individual operating components of the systems) may not be described in
10   detail herein. Furthermore, the connecting lines shown in the various figures contained herein are intended to represent exemplary functional relationships or physical couplings between the various elements. It should be noted that many alternative or additional functional relationships or physical connections may be present in a practical system.

15   It will be appreciated that many applications of the present invention could be formulated. One skilled in the art will appreciate that the network may include any system for exchanging data or transacting business, such as the Internet, an intranet, an extranet, WAN, LAN, satellite communications, or the like. The users may interact with the system via any input device such as a keyboard,
20   mouse, kiosk, personal digital assistant, handheld computer (e.g., Palm Pilot®), cellular phone or the like. Similarly, the invention could be used in conjunction with any type of personal computer, network computer, workstation, minicomputer, mainframe, or the like running any operating system such as any

version of Windows, Windows NT, Windows 2000, Windows ME, Windows 98, Windows 95, MacOS, OS/2, BeOS, Linux, UNIX, or the like. Moreover, although the invention is frequently described herein as being implemented with TCP/IP communications protocols, it will be readily understood that the invention could

5   also be implemented using IPX, Appletalk, IP-6, NetBIOS, OSI or any number of existing or future protocols. Moreover, the system contemplates the use, sale or distribution of any goods, services or information over any network having similar functionality described herein.

As will be appreciated by one of ordinary skill in the art, the present

10   invention may be embodied as a method, a data processing system, a device for data processing, or a computer program product. Accordingly, the present invention may take the form of an entirely software embodiment, an entirely hardware embodiment, or an embodiment combining aspects of both software and hardware. Furthermore, the present invention may take the form of a

15   computer program product on a computer-readable storage medium having computer-readable program code means embodied in the storage medium. Any suitable computer-readable storage medium may be utilized, including hard disks, CD-ROM, optical storage devices, magnetic storage devices, or the like.

Communication between the parties to the transaction and the system of

20   the present invention is accomplished through any suitable communication means, such as, for example, a telephone network, Intranet, Internet, point of interaction device (point of sale device, personal digital assistant, cellular phone, kiosk, etc.), online communications, off-line communications, wireless

communications, or the like. One skilled in the art will also appreciate that, for security reasons, any databases, systems, or components of the present invention may consist of any combination of databases or components at a single location or at multiple locations, wherein each database or system

5 includes any of various suitable security features, such as firewalls, access codes, encryption, de-encryption, compression, decompression, or the like.

The present invention is described herein with reference to block diagrams and flowchart illustrations of methods, apparatus (*e.g.*, systems), and computer program products according to various aspects of the invention. It will

10 be understood that each functional block of the block diagrams and the flowchart illustrations, and combinations of functional blocks in the block diagrams and flowchart illustrations, respectively, can be implemented by computer program instructions. These computer program instructions may be loaded onto a general purpose computer, special purpose computer, or other programmable

15 data processing apparatus to produce a machine, such that the instructions which execute on the computer or other programmable data processing apparatus create means for implementing the functions specified in the flowchart block or blocks.

These computer program instructions may also be stored in a computer-

20 readable memory that can direct a computer or other programmable data processing apparatus to function in a particular manner. The computer program instructions may also be loaded onto a computer or other programmable data processing apparatus to cause a series of operational steps to be performed on

the computer or other programmable apparatus to produce a computer-implemented process such that the instructions which execute on the computer or other programmable apparatus provide steps for implementing the functions specified in the flowchart block or blocks.

5   Accordingly, functional blocks of the block diagrams and flowchart illustrations support combinations of means for performing the specified functions, combinations of steps for performing the specified functions, and program instruction means for performing the specified functions. It will also be understood that each functional block of the block diagrams and flowchart

10   illustrations, and combinations of functional blocks in the block diagrams and flowchart illustrations, can be implemented by either special purpose hardware-based computer systems which perform the specified functions or steps, or suitable combinations of hardware and computer instructions.

The computing units may be connected with each other via a data

15   communication network. The network may be a public network and assumed to be insecure and open to eavesdroppers. In the illustrated implementation, the network may be embodied as the Internet. In this context, the computers may or may not be connected to the Internet at all times. It is noted that the network may be implemented as other types of networks, such as an interactive

20   television (ITV) network.

The present invention is directed to a transparent or substantially transparent method of accessing data and operating programs across various computing platforms. More particularly, the present invention allows programs

written for various platforms to interact with each other. For purposes of illustration only, the configuration of particular embodiments will be shown to describe the invention. It should be understood, however, that the scope of the invention is not limited by such illustrations, as it is the claims at the end of this

5   patent application that determine the scope of the invention.

In an exemplary embodiment, the system includes a bi-directional interoperability framework between diverse platforms, such as, for example, between WAS/SUN/JAVA and IIS/NT/COM applications. The system is a consolidated package for diverse platforms to interoperate and transfer data.

10   The invention allows the development of components on, *e.g.*, the WAS/SUN/Java platform, while at the same time, allowing applications on, *e.g.*, the IIS/NT/COM platform to reuse the services/components built on WAS/SUN/Java. Further, the cross-platform services, which typically included a separate effort on WAS/SUN/Java, as well as the IIS/NT/COM side, can now be

15   offered through an interoperability channel, thus alleviating the need for two separate development efforts. Moreover, the invention can be delivered as an enterprise infrastructure solution within an organization, thereby allowing new applications to reuse the business functions/components already built on both WAS/SUN/Java and IIS/NT/COM platforms.

20   An overview of an exemplary embodiment of the present invention is presented in **Figure 1**. NT/COM/IIS system 112 communicates with client 110 and various programs 114. In a similar manner, WAS/SUN/JAVA system 122 communicates with client 120 and various programs 124. In a traditional system,

system 112 is not able to communicate or share data with system 122. However, an embodiment of the present invention involves the use of an interoperability component 102 to allow the systems to communicate with each other and exchange information.

5   **Figure 2** presents further details of the structure of an exemplary embodiment of the present invention. It may be desired to exchange information between an IIS/NT/COM component 202 and a WAS/SUN/JAVA component 204. COM component 202 may be configured to communicate a request to a transmission component 212. Transmission component 212 may be configured

10   to transmit the request to a request component 214. Request component 214 can build and format a request stream and transmit the stream back to transmission component 212. Transmission component 212 is configured to be in communication with JAVA object 250 and to transmit the request stream. JAVA object 250 is configured to translate the request stream and obtain the

15   requested data or perform the requested command via communication with WAS/SUN/JAVA component 204. The information can then be transmitted back to transmission component 212 through JAVA object 250, which may translate the response and transmit the response to COM component 202. In an exemplary embodiment, the information communicated between transmission

20   component 212 and JAVA object 250 is through mutually agreed eXtensible Markup Language ("XML") messages, transmitted via HyperText Transfer Protodcol ("HTTP") or Secure HyperText Transfer Protocol ("HTTPS"), using a version of Secure Sockets Layer ("SSL").

Figure 9 presents a flowchart illustrating an exemplary overview of the operation of an embodiment of the present invention. A request is made at a first computing platform (step 902). The request is translated into a predetermined format that is readable by both the first computing platform and

5   the second computing platform (*e.g.*, XML) (step 904). The request is then transmitted from the first computing platform to the second computing platform (step 906). At the second computing platform, the request is translated from the predetermined format into a format usable by the second computing platform (step 908). The second computing platform then processes the request and

10   either acquires the requested data or performs the requested task (step 910). The result is then translated into the predetermined format (step 912), and transmitted to the first computing platform (step 914). The first computing platform translates the result from the predetermined format (step 916) and displays the result or otherwise makes use of the result (step 918).

15   An exemplary embodiment of the present invention may be written in Microsoft's Visual Basic language or other suitable language. For example, the Logical Class Diagram 700 of **Figure 7**, illustrates various classes that can be used in a Visual Basic embodiment. The *InteropTransmission* class 702 may be configured to interact with an *InteropRequest* class 704 and an

20   *InteropParameter* class 706. It should be noted that the *InteropTransmission* class 702 is responsible for the entire transmission (the posting of XML messages over HTTP or HTTPS) to an external system with which the communication is desired. *InteropTransmission* class 702 may be configured to

send multiple *InteropRequests* in a single transmission. A request may specify a single request that may have multiple input/output parameters.

The *InteropTransmission* class is configured to transmit and receive data from an external system and is responsible for maintaining the information used to communicate with the external system. Such information may include the server name (*i.e.*, where the Interoperability Component is located on the local server), the Uniform Resource Locator ("URL") of the external system's Interoperability Component, and any access information required by the external system, such as user identification and a password.

The *InteropRequest* class is configured to produce the XML required to build a request to be sent to the external system. It is also configured to convert obtain the reply and convert it into the format expected by the output parameters for the method and returning these in the output parameters of its parameters collection.

The *InteropRequest* class may contain the following variables: 1) *BeanName*, which contains the name of the bean owning the service requested; 2) *MethodName*, which contains the name of the method in the bean implementing the service; 3) *Parameters*, which contains an ordered collection of the parameters expected or returned by the method; and 4) *XMLResponse*, which contains the XML response result from the transmission of the request. The class may also contain a method called *MtdGetXML*, which produces the XML for the request. The method *MtdGetXML* may be configured to use its InteropParameters to build its XML request stream, returning it to its owner

InteropTransmission object. This method may call the method *InteropParameter::getXML*, which obtains the XML representation of each parameter.

The *InteropParameter* class is responsible for maintaining the value, type, and direction of a single parameter. The *InteropParameter* class also transforms the parameter to and from XML format. The class may contain various variables. For example, as illustrated in **Figure 7**, there may be variables containing the name of the parameter, the type of parameter, the value of the parameter, the direction (input or output) of the parameter, and the XML representation of the parameter. In a preferred embodiment, when the value of the XML string is set, the Value property of the property is updated from the contents of the XML stream. When this value is retrieved, the string is built from the current attributes of the parameter.

Various methods (code used to affect the object with which the method is associated) may also be present in a Visual Basic embodiment of the present invention. In the *InteropTransmission* class, there may be a method called *mtdPostRequests* which uses *InteropRequests* to build an XML request stream, send the stream via HTTPS, and wait for the response. The response can then be parsed to separate the responses for each request. The XMLResponse property of each *InteropRequest* is set accordingly.

An exemplary embodiment of the InteropTransmission class contains a variable called Requests that contains a collection of requests that are to be sent in a single transmission to the external system. It may also contain a

method titled MtdPostRequests that posts the requests to the external system. A call to the method may be of the following format:

MtdPostRequests(DestinationServerName, URL, AccesInformation)

where *DestinationServerName* contains the address where the

5   interoperability component resides; *URL* contains the URL specifying the external system's interoperability component; and *AccessInformation* may contain an array of information required to access the external system. For example, *AccessInformation* may contain a User Id and Password that is needed to access the external system. In the alternative, *AccessInformation* may contain

10 an array of strings containing the access information needed to access the external system.

The MtdPostRequests method will use *InteropRequests* to build its XML request stream, send it via HTTPS, and wait for the response. In addition, other methods can be called from this method, including *InteropRequest::mtdGetXML,*

15 which obtains the XML Request stream for each request; and *InteropRequest::put_XMLResponse*, which sets the XML Response stream for the request.

For purposes of illustration, an exemplary embodiment showing an interconnection between two entities, SDP and IT, will now be described. It

20 should be understood that, for the illustrated embodiments, SDP is an exemplary entity that uses a NT/IIS/COM platform and IT is an exemplary entity that uses a WAS/SUN/JAVA platform.

With reference to **Figure 3**, the steps performed in an exemplary use of the system is as follows. Figure 3, in particular, illustrates IT access to SDP services. A customer using an IT website makes a request that causes a Java Server Page ("JSP") on the IT Web Server to be run (step 302). Because the

5   request cannot be satisfied using JSP computing resources, the JSP on the IT Web Server determines that it must obtain the data requested by the client from the SDP system, so it invokes an Internet Server Application Programming Interface ("ISAPI") extension Dynamic Load Library ("DLL") on the SDP Web Server using the URL for the extension. The information required by the

10   extension to run the requested method is translated into the XML stream that will identify the business request and is sent via HTTP (step 304). The extension then invokes a SDP Microsoft Transaction Server ("MTS") COM object to perform the necessary functions to fulfill the request (step 306). The SDP COM object performs the requested business work, which may involve gathering data

15   from a back-end system, and presenting the data in the desired format (step 308). The SDP COM object returns the resulting data to the ISAPI extension (step 310). The ISAPI extension converts the data to XML and passes it to the IT Web Server via a HTTP response (step 312). The XML data is parsed on the IT side and sent to the JSP. The JSP obtains the data it requires to build its

20   Hypertext Markup Language ("HTML") page. The JSP then serves the resulting HTML page to the client browser (step 314). It should be pointed out that, in an exemplary embodiment, steps 306-312, listed above, are performed by the

external server, while steps 302, 304, and 314 are performed at the originating, IT server.

With reference to **Figure 4**, the steps of an exemplary use of the system for SDP access to IT services is as follows. A Customer Service Representative
5 ("CSR") at the service center makes a request that causes an Active Server Page ("ASP") on the SDP Web Server to be run (step 402). The ASP on the SDP Web Server determines that it must obtain the data requested by the CSR from the IT System, so it invokes a servlet on the IT Web Server using the URL for the required servlet. The information required by the servlet to run the
10 requested method is translated to an XML format and is sent via HTTPS (step 404). The servlet invokes the IT business component to perform the tasks to supply the requested information (step 406). The IT business component then performs the requested tasks (step 408), then returns the resulting data to the servlet (step 410). The servlet translates the data to XML and passes it to the
15 SDP Web Server through HTTP or HTTPS (step 412). The data is then sent to the first server/platform (step 413). The ASP parses the XML data, converting the data to a readable format and obtaining the data it requires to build its page. The ASP then serves the resulting page to the client browser. It should be noted that steps 406-413 above are performed by the external server.

20 The operation of a further embodiment of the present invention will now be presented. **Figure 5** illustrates the operation of an embodiment of the invention when SDP requires account information available through IT service. An ASP on SDP activates the Business Process Manager ("BPM") responsible

for providing the account information (step 502). BPM creates an instance of an *InteropTransmission* class (step 504). The *InteropTransmission* class is created and creates an empty collection of *InteropRequest* (step 506). BPM transmits an *InteropRequest* to the *InteropTransmission* class (step 508). The

5 *InteropTransmission* class creates the request, adds it to its collection, and returns a reference to the new request to the BPM (step 510). The BPM sets the Request's bean name and method (step 512). The BPM adds the input and output parameters to the request, via the *AddParameter* method, assigning the value property to all parameters that are in or in/out (step 514). The BPM calls

10 the *InteropTransmission::mtdPostRequests* method, passing it the required information (step 516). The *InteropTransmission* class obtains the XML for each request via the *InteropRequest::mtgGetXML* method for each request to be sent in the transmission, using the XML strings to build the single XML stream to be sent in the transmission (step 518). The *InteropTransmission* builds an XML

15 stream, using *InteropParameters:: mtdGetXML* to get the XML representation of each in or in/out parameter (step 520). *InteropRequest* returns the XML stream to the *InteropTransmission* class (step 522). The *InteropTransmission* class builds the entire stream and sends it to the external system via HTTPS (step 524). The *InteropTransmission* class receives XML response (step 526) and

20 obtains the XML response for each request, assigning the XMLResponse property of each request accordingly (step 528). The *InteropRequest* object parses the XML Response, matching output parameters in the stream to the corresponding output parameters in the collection of parameters (step 530).

In an exemplary embodiment, for each such parameter: 1) the *InteropRequest* class sets the XML property of the parameter (step 532); 2) the *InteropParameter* class parses the XML stream to obtain the value returned, setting the Value property accordingly (step 534); 3) the *InteropParameter* class

5   returns the result to the *InteropRequest* (step 536). The *InteropRequest* class then returns to the *InteropTransmission* class (step 538). The *InteropTransmission* class returns to the BPM (step 540) and the BPM examines the Parameters property of each *InteropRequest* to obtain the desired output values to return to the ASP page (step 542).

10   With reference to **Figure 6**, the operation of an embodiment of the present invention when SDP receives a request from an IT client will now be described. SDPServlet receives the request from *InteropTransmission* on SDP Server (step 602). *SDPServlet* redirects the request to Component Provider (step 604). Component Provider passes the request (in XML format) to

15   *SDPMessage* class to get the request method (step 606). *SDPMessage* class invokes *SDPXMLUtility* to extract all the parameters from input XML message and creates an object *SDPRequest* that contains all input parameters (step 608). *SDPMessage* returns *SDPRequest* object to Component Provider (step 610). Component Provider then passes the *SDPRequest* object to *SDPSecurity* Class

20   to extract Security token and a Global Unique Identifer ("GUID") from input Cookie String (step 612). GUID is a unique identifier that uniquely identifies the user. Each GUID can be logged to form a record as to the particular transactions performed by each user.

SDPSecurity class returns *SDPRequest* (which now contains the Security Token and GUID also) object to Component Provider (step 614). Component Provider passes *SDPRequest* object to *ComponentAccessor* Class (step 616). *ComponentAccessor* identifies the Component that needs to be invoked with the

5 help of Parameter (step 618). *ComponentAccessor* class creates the instance of a *ComponentHandler* class for the Component to be invoked (step 620). *ComponentAccessor* invokes the *getITR* response method of *ComponentHandler* class, and passes the *SDPRequest* Object to it (step 622). *ComponentHandler* class obtains all the parameters required from *SDPRequest*

10 object and invokes the component (step 624). *ComponentHandler* creates an *ITResponse* class based on the response received from the Component and returns it to *ComponentAccessor* (step 626). *ComponentAccessor* returns *ITResponse* object to *ComponentProvider* (step 628). *ComponentProvider* passes the *ITResponse* object to *SDPMessage* to convert the response to the

15 required format (step 630). *SDPMessage* uses *SDPXMLUtility* to create Response Method in XML format (step 632). *SDPMessage* returns Response String to *ComponentProvider* (step 634). *ComponentProvider* returns the response XML String to *SDPServlet* (step 636). *SDPServlet* returns the response XML String to *InteropTransmission* on SDP Server (step 638).

20 **Figure 8** illustrates the operation of a system of an exemplary embodiment of the present invention. A client 802 accesses a COM server 810, possibly through the Internet, then client 802 accesses an Active Server Page ("ASP") 812. Such access may be controlled by SSO 806 and policy server 804.

Via an internal COM link 814, client 802 has access to business component 816, which communicates via internal COM link 818 to an interoperability component 820, which formulates and transmits an XML request using HTTPS.

Web Server 840 receives the XML request thorough link 830, translates, and transmits the request to WAS server 850. Within WAS server 850 is a servlet 852, component provider 854, XML utility 856, and component accessor 858. Those components process the request and transmit the request to a handler 860. Handler 860 may perform many functions. For example, handler 860 may determine the balance summary of a requested account. However, it should be understood that such a handler can configured to process a variety of different functions. Handler 860 translates the request into a command bean 862, which interfaces, through link 864 with a command bean 874 located within another WAS server 870. Command bean 874 is configured to access sample service 872 and process the request of user 802. The result is then transmitted back through the above-described component to Web Server 840, where the result is transmitted via HTTPS as an XML stream. The XML stream is processed in COM server 810 and the result is eventually transmitted to client 802.

The above described process may be substantially seamless to client 802 such that client 802 has minimal knowledge or is unaware of the access to a different server. In such a manner, client 802 can access data located on WAS server 870 which was formerly unavailable or limited access due to the incompatible computing systems.

An exemplary XML record layout of an exemplary embodiment will now be presented. The following XML code represents an exemplary Document Type Definition ("DTD") used an embodiment of the present invention:

```
     <!DOCTYPE ITINPUTDATA [
5    <!ELEMENT SysID (#PCDATA)>
     <!ELEMENT DebugTrk (#PCDATA)>
     <!ELEMENT PerfTrk (#PCDATA)>
     <!ELEMENT ITUserID (#PCDATA)>
     <!ELEMENT ITPswd (#PCDATA)>
10   <!ELEMENT SDPUserID (#PCDATA)>
     <!ELEMENT Component (ComponentName, Method Type, (Parameter))>
     <!ELEMENT ComponentName (#PCDATA)>
     <!ELEMENT MethodType (#PCDATA)>
     <!ELEMENT Parameter (#PCDATA)
15   ]>
```

Utilizing the above DTD, the following XML code is a sample request for a balance summary in an exemplary embodiment where financial data is exchanged among different platforms:

```
     <ITINPUTDATA>
20   <SysID>IT</SysID>
     <DebugTrk>0</DebugTrk>
     <PerfTrk>0</PerfTrk>
     <ITUserID>EC9396B</ITUserID>
     <ITPswd>1234abcd</ITPswd>
25   <Component
          <ComponentNm>
               BalanceSummary
          </ComponentNm>
          <MethodType>
30             BalanceSummaryPropertiesCommand
          </MethodType>
          <Parameter>
               34566742211232
          </Parameter>
35   </Component
     </ITINPUTDATA>
```

An exemplary response XML stream (for a different request) is as follows:

```
<ITRESPONSE>
<SysID>IT</SysID>
<ITRespType>0</ITRespTyp>
<ITResponseContent>
        <Data>
        <PropertyTable>
                <PropertyRec>
                        <Property>
                                <PropNm>Type_platinum</PropNm>
                                <PropVal>True</PropVal>
                        <Property>

                                .

                        <Property>
                                <PropNm>AccountStatus</PropNm>
                                <PropVal>ACTIVE</PropVal>
                        <Property>
                </PropertyRec>
                <PropertyRec>
                        <Property>
                                <PropNm>Type_platinum</PropNm>
                                <PropVal>True</PropVal>
                        <Property>

                                .

                        <Property>
                                <PropNm>AccountStatus</PropNm>
                                <PropVal>ACTIVE</PropVal>
                        <Property>
                </PropertyRec>
        </PropertyTable>
        </Data>
</ITResponseContent>
</ITRESPONSE>
```

The above-described approach has many benefits. For example, the use

of a reliable, proven communication protocol eases the introduction of

embodiments of the present invention. In addition, it is easy to implement, using

ASP in SDP and JSP in IT. Furthermore, no additional software is required for

either platform and only a small amount of code on both sides, to handle the

requests/responses, is needed.

The interoperability framework has numerous other benefits including, for example, reduced future application development cost, reduced future application maintenance cost, reduced investment in duplicate technology and software, effective utilization of available infrastructure, reduced cost of development and operations, improved speed of development, leveraging existing cross application/cross platform resources, and improved inter-group communication and awareness. The invention also enables standard information to be provided to customers irrespective of computer platform and efficient communication across business/technology groups.